www.conairnet.com

USERGUIDE

SYSTEM CONFIGURATION
for S900II robots
Software Version 1.0

⚠ **WARNING - Reliance on this Manual Could Result in Severe Bodily Injury or Death!**
This manual is out-of-date and is provided only for its technical information, data and capacities. Portions of this manual detailing procedures or precautions in the operation, inspection, maintenance and repair of the product forming the subject matter of this manual may be inadequate, inaccurate, and/or incomplete and cannot be used, followed, or relied upon. Contact Conair at info@conairgroup.com or 1-800-654-6661 for more current information, warnings, and materials about more recent product manuals containing warnings, information, precautions, and procedures that may be more adequate than those contained in this out-of-date manual.

Logo definitions :

- Warning, risk
- Sepro robotique innovations
- What to do ?
- Document evolutions
- Handy hints
- Example
- V xx Software innovation

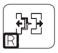
I – MEMORY

I – 1. Accessing the memory

After accessing ”Memory Management” by pressing [Memo_M] (programming menu), pressing the [M_Read] key gives access to the read (or modification) function of the user and system RAM or EEPROM memory (at the address of the memory box by default if necessary).

The address of the area at which reading is to begin is given in hexadecimal (0 to F) using the numerical keypad and the first row of alphanumerical keys of the keyboard.

Certain areas are directly accessible from the keyboard :

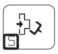
 : beginning of the PRG editing area (0 x 006 430).

 : beginning of the PLC editing area (0 x 009 430).

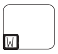
 : beginning of the program storage in RAM area (0 x 00B 300).

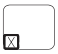
 : beginning of the MODULE where the programs are stored (0 x 800 000).

 : transfer buffer PRG E17.

 : robot serial number in RAM.

 : RAM access password.

 : beginning of parameters in RAM.

 : beginning of the faults 200 to 204 message table in RAM.

For example : to access the beginning of the program storage area, the procedure is as follows :

[Memo_M] –> [M_Read] –> [Address] –> 

* **The keys** :

- ▶ [+] or [–] to change addresses 2 by 2.
- ▶ [↑] or [↓] to change addresses 10 by 10 (hexadecimal).
- ▶ [PG DN] or [PG UP] to change addresses 100 by 100 (hexadecimal).

* **The function keys** F1 to F5 :

- ▶ **[Address]** to change the address.
- ▶ **[Modif]** to change the contents of the memory area displayed (word).
- ▶ **[Search]** to search for a particular word (e.g. : FA1B)
- ▶ **[Print]** to print the memory contents from the displayed address (in order to search for the incorrect instructions which will be printed as ????).
- ▶ **[StopPr]** to stop sending the memory contents to the printer.

Note : To access the modification function, a password is necessary which remains valid as long as the user does not exit the “M_Read” procedure. Certain critical system areas cannot be read and all requests to modify them will be rejected.

By default, the value given after modification request is 0 x FFFF (useful to delete words in the memory).

As for the other functions, the EXIT key is used to abandon a request or to exit the procedure.

I – 2. Memory areas

I – 2. 1.Data saved in RAM (512 K x 8) 0 to 7 FFFF

Address in Hexadecimal	Contents
00000 027FF	Variables used by Philips (BOOT)
02800 0A4FF	“Fixed” SEPRO variables, see table below for details of the variables
0A500 0B2FF	SEPRO parameters in RAM
0B300 2A6FF	PRG storage area (128 K × 8)
2A700 37FFF	SEPRO variables / work tables
38000 57FFF	Temporary transfer area (128 K x 8)
58000 7FFFF	Piles and heaps used by the ERM kernel

02800	En Ordre = RAM contents correct indicator (GIRLAFRIDOU).
02810	Bit_U_S = System and user bits table.
02890	Bit_Tpo = PLC timer bits table.
028A0	Imag_S = Images of the 255 ON/OFF outputs.
029A0	Imag_E = Image of the 255 ON/OFF inputs.
02AA0	Word_U = User words table (16–bit WORD).
02AE0	Word_S = System words table (see Programming Level 2 manual for description).
02B20	Tpo_Aut = PLC timers table.
02B40	Compt = Counters table (standard and stacking).
04AA0	Pile_Def = Pile of historic faults.
04BC0	Comptime = Times basic counter.
04BC4	Dir_RAM = PRG / PLC directory in editing area.
04C04	Dir_PP = PRG directory in save area.
05254	Dir_PLC = PLC directory in save area.
05710	Mod_PP = PRG directory in the module.
05D60	Mod_PLC = PLC directory in the module.
0621C	Tab_temps = Robot times table.
06230	WWord_U = Double words table (32 bits).
06430	Ram_PP = PRG editing area.
09430	Ram_PLC = PLC editing area.

I – 2. 2.Program addressing in memory

The PRG and PLC programs are stored in the RAM memory, starting from the address 0xB300.

The maximum length of a PRG is 12286 bytes ; 4096 bytes for a PLC.

This area reserved for the permanent storage varies depending on the option 32 to 128 Kbytes.

So that it remains compatible with previous software versions, the RAM if formatted with 0xFFFF like an EEPROM. This formatting is carried out when the robot is first started up (for the 128 Kbytes) or when the memory is totally set to 0 [RsMEM] (on the size provided for in the options)

The parameters are stored in FLASHPROM at the address 0xF10E0000. An image of this address is stored in RAM at the address 0xA500. The length of the parameters is fixed at 2800 bytes.

The “SAP message” file is stored in FLASHPROM at the address 0xF10E1200. Its length is fixed at 4590 bytes.

The programs, parameters and SAP messages are transferred via a temporary buffer of 12286 bytes at the address 0x38000. (This buffer can be extended to 128 Kbytes).

I – 2. 3.Data in Flashprom (1 M x 8) F10 00000 to F10 FFFFF

Block number	Address in Hexadecimal	Contents
1st block	F10 00000 F10 0FFFF	ERM kernel + SEPRO program
	F10 10000 F10 1FFFF	SEPRO code (1)
2nd block	F10 20000 F10 3FFFF	SEPRO code (2)
3rd block	F10 40000 F10 5FFFF	SEPRO code (3)
4th block	F10 60000 F10 7FFFF	SEPRO code (4)
5th block	F10 80000 F10 9FFFF	SEPRO code (5)
6th block	F10 A0000 F10 BFFFF	Reserved for extension of SEPRO code

Block number	Address in Hexadecimal	Contents
7th block Messages	F10 C0000 F10 CEBEF	Messages in language 1
	F10 CEBF0 F10 DD7DF	Messages in language 2
	F10 DD7E0 F10 DE7EF	Font robot 1
	F10 DE7F0 F10 DF7FF	Font robot 2
	F10 DF800 F10 DF9FF	Code converter table IMM 1
	F10 DFA00 F10 DFBFF	Code converter table IMM 2
	F10 DFC00 F10 DFDFF	Code converter table Printer 1
	F10 DFE00 F10 DFFFF	Code converter table Printer 2
8th block Parameters and SAP	F10 E0000 F10 E0DFF	SEPRO parameters
	F10 E1200 F10 E2256	SAP messages
	F10 E2400 F10 FFFFF	Reserved for SEPRO

I – 3. Specific information

These are directly accessed using the Memory Read function followed by the request [Address] and a letter :

– to access the memory area containing the passwords.

– to access the memory area containing the serial number and the type of robot.

Address	15	0	
B2A0	00	00	Password to access edition (....)
B2A2	00	00	
B2A4	00	00	Password to access parameters (....)
B2A6	04	D2	
B2A8	00	00	Password to access maintenance (....)
B2AA	00	00	
B2AC	00	00	Password to block the modes (....)
B2AE	00	00	
B2B0	00	00	Password to block the selection of the PRG N° to be executed (....)
B2B4	00	00	
...			
B2E0			Operating time.
B2E2			
B2E4			Operating time in automatic.
B2E6			
B2E8	00	00	Robot serial number : E.g. 1024
B2EA	04	00	
B2EC	00	35	Robot type :
B2EE	73	98	

E.g. 350 BB (000) –> 3503000–D –> 357398–H

- 350 → Model
- BB → Type
- (000) → Specific

Type:

0 BX
1 BY
2 BZ
3 BB
4 BC
5 AX
6 AY
7 AZ

II – INSTRUCTION CODES

II – 1. Part programs

Type of Instruction	Display	Codop (hexadecimal)	Examples
ACTION	ACT 00 (to 99) *	A000 [oper. 16 bits] → Action No.	A000000C = ACT12
OUTPUT	OUT 000 (to 255) *	A001 [oper. 16 bits] → Output No.	A0010050 = OUT080
INPUT Normal	IN 000 (to 255)	A002 [oper. 16 bits] → Input No.	A002000A = IN010
INPUT Reverse	IN/000 (to 255)	A003 [oper. 16 bits] → Input No.	A0030020 = IN/032
TIMER	TIME 001 to 999	A004[oper.4bits]0[oper.11bits] → SAP marker No. / Value in 1/10s	A004000A = TIME010 A004300A = TIME010 Marker P03
	TIME W_00 à 15	A004 0000 1 [oper.11bits] → Word No.	A004080A = TIMEW10 A004080F = TIMEW15
BIT	BIT 000 (to 127)	A005 [oper. 16 bits] → Bit No.	A0050063 = BIT 99
	/ BIT 000 (to 127)	A006 [oper. 16 bits]	A006007D = BIT 127

* The actions and outputs replaced by text (e.g.: `part grip 1`) keep the same CODOP

Type of Instruction	Display	Codop (hexadecimal)	Examples
FUNCTIONS (FUNC)			
SPEED in % of the parametered speed	VEL.X 001 to 100 VEL.Y 001 to 100 VEL.Z 001 to 100 VEL.B 001 to 100 VEL.C 001 to 100	B000[oper.4bits][oper.12bits] B001[oper.4bits][oper.12bits] B002[oper.4bits][oper.12bits] B003[oper.4bits][oper.12bits] B004[oper.4bits][oper.12bits] ↓ SAP marker N° ↓ Value in 1/10s	B0000062 = VEL.X 098 B001000A = VEL.Y 010 B0020012 = VEL.Z 018 B0030064 = VEL.B 100 B004A032 = VEL.C 050 Marker P10
	VEL.X WW_*nn VEL.Y WW_*nn VEL.Z WW_*nn VEL.B WW_*nn VEL.C WW_*nn *(nn = 00 to 55 and 66 to 67)	B050 0000 [oper.12bits] B051 0000 [oper.12bits] B052 0000 [oper.12bits] B053 0000 [oper.12bits] B054 0000 [oper.12bits] ↓ Word No.	B0500042 = VEL.X WW066 B0510043 = VEL.Y WW067 B0520042 = VEL.Z WW066 B0530042 = VEL.B WW066 B0540043 = VEL.C WW067
ACCELERATION in % of the parametered acceleration	ACC.X 001 to 100 ACC.Y 001 to 100 ACC.Z 001 to 100 ACC.B 001 to 100 ACC.C 001 to 100	B010 [oper. 16 bits] B011 [oper. 16 bits] B012 [oper. 16 bits] B013 [oper. 16 bits] B014 [oper. 16 bits] ↓ Value in %	B010000F = ACC.X 015 B0110064 = ACC.Y 100 B0120044 = ACC.Z 068 B0130005 = ACC.B 005 B0140032 = ACC.C 050
Master MOVEMENT	MASTER.X MASTER.Y MASTER.Z MASTER.B MASTER.C	B030 B031 B032 B033 B034	
IMPRECISION	IMP.X IMP.Y IMP.Z IMP.B IMP.C	B040 B041 B042 B043 B044	

Type of Instruction	Display	Codop (hexadecimal)	Examples
MOTORIZED MOTIONS			
SLOW APPROACH in % of the maximum parametered speed	SLA.X 001 to 100 SLA.Y 001 to 100 SLA.Z 001 to 100 SLA.B 001 to 100 SLA.C 001 to 100	B020 [oper. 16 bits] B021 [oper. 16 bits] B022 [oper. 16 bits] B023 [oper. 16 bits] B024 [oper. 16 bits] Value in %	B0200026 = SLA.X 026 B0210034 = SLA.Y 034 B0220090 = SLA.Z 090 B0230100 = SLA.B 100 B0240010 = SLA.C 010
LINEAR			
ABSOLUTE *(Numerical operands)*	X.ABS_L distance Y.ABS_L distance Z.ABS_L distance B.ABS_L distance C.ABS_L distance	C000[oper.8bits][oper.24bits] C001[oper.8bits][oper.24bits] C002[oper.8bits][oper.24bits] C003[oper.8bits][oper.24bits] C004[oper.8bits][oper.24bits]	C00000000664=X.ABS.L00163.6 C001000F423F=Y.ABS.L99999.9 C00200000320=Z.ABS.L00080.0 C0030000003F=B.ABS.L00006.3 C0040000050C=C.ABS.L00150.0
STACKING	X.STK_L distance Y.STK_L distance Z.STK_L distance B.STK_L distance C.STK_L distance	C010[oper.8bits][oper.24bits] C011[oper.8bits][oper.24bits] C012[oper.8bits][oper.24bits] C053 C054	C01000008ACF=X.STK.L03453.5 C01100030DE3=Y.STK.L20016.3 C01200000159=Z.STK.L00034.5 Reserved for general STKs Absolute distances from the header
RELATIVE	X.REL_L distance Y.REL_L distance Z.REL_L distance B.REL_L distance C.REL_L distance	C020[oper.8bits][oper.24bits] C021[oper.8bits][oper.24bits] C022[oper.8bits][oper.24bits] C023[oper.8bits][oper.24bits] C024[oper.8bits][oper.24bits]	C020800000A0=X.REL.L–0016.0 C021000000A0=Y.REL.L–0016.0 C0228001869F=Z.REL.L–9999.9 C02300002706=B.REL.L+0999.9 C0240000000A=C.REL.L+0001.0
CHECKING	X.CTL_L distance Y.CTL_L distance Z.CTL_L distance B.CTL_L distance C.CTL_L distance	C030[oper.8bits][oper.24bits] C031[oper.8bits][oper.24bits] C032[oper.8bits][oper.24bits] C033[oper.8bits][oper.24bits] C034[oper.8bits][oper.24bits] SAP marker No. Distance in 1/10 mm	C03000000664=X.CTL.L00163.6 C031000F423F=Y.CTL.L9999.9 C03200000320=Z.CTL.L00080.0 C0330000003F=B.CTL.L00006.3 C0340500050C=C.CTL.L00150.0 Marker P05
ROTATING			
ABSOLUTE *(Numerical operands)*	X.ABS_R Angle Y.ABS_R Angle Z.ABS_R Angle B.ABS_R Angle C.ABS_R Angle	C100[oper.8bits][oper.24bits] C101[oper.8bits][oper.24bits] C102[oper.8bits][oper.24bits] C103[oper.8bits][oper.24bits] C104[oper.8bits][oper.24bits]	C10000000664=X.ABS.R00163.6 C101000005DC=Y.ABS.R00150.0 C10200000320=Z.ABS.R00080.0 C1030000003F=B.ABS.R00006.3 C10400000159=C.ABS.R00034.5
STACKING	X.STK_R Angle Y.STK_R Angle Z.STK_R Angle	C110[oper.8bits][oper.24bits] C111[oper.8bits][oper.24bits] C112[oper.8bits][oper.24bits]	C11000008ACF=X.STK.R03453.5 C11100030DE3=Y.STK.R20016.3 C11200000159=Z.STK.R00034.5
RELATIVE	X.REL_R Angle Y.REL_R Angle Z.REL_R Angle B.REL_R Angle C.REL_R Angle	C120[oper.8bits][oper.24bits] C121[oper.8bits][oper.24bits] C122[oper.8bits][oper.24bits] C123[oper.8bits][oper.24bits] C124[oper.8bits][oper.24bits]	C12000000384=X.REL.R+90.0 C12180000320=Y.REL.R–90.0 C12200000320=Z.REL.R+80.0 C12380000159=B.REL.R–34.5 C1240000003F=C.REL.R+06.3

Type of Instruction	Display	Codop (hexadecimal)	Examples
CHECKING	X.CTL_R Angle Y.CTL_R Angle Z.CTL_R Angle B.CTL_R Angle C.CTL_R Angle	C130[oper.8bits][oper.24bits] C131[oper.8bits][oper.24bits] C132[oper.8bits][oper.24bits] C133[oper.8bits][oper.24bits] C134[oper.8bits][oper.24bits] SAP Marker No. / Angle in 1/10 deg.	C13000000664=X.CTL.R00163.6 C131000F423F=Y.CTL.R9999.9 C13200000320=Z.CTL.R00080.0 C1330000003F=B.CTL.R00006.3 C1340000050C=C.CTL.R00150.0
TEACHING	Teach Previous instruction	C [oper.8bits]AAAAAA Instruction code / SAP marker No.	C01000AAAAAA=X.STK.LTeach C10200AAAAAA=Z.ABS.RTeach
<u>MOTORIZED</u> <u>MOTIONS</u> (cont'd)			
<u>LINEAR</u>			
ABSOLUTE *(Words)*	X.ABS_L WW *nn Y.ABS_L WW *nn Z.ABS_L WW *nn B.ABS_L WW *nn C.ABS_L WW *nn	C200 [oper. 16 bits] C201 [oper. 16 bits] C202 [oper. 16 bits] C203 [oper. 16 bits] C204 [oper. 16 bits]	C200000A = X.ABS.L WW10
STACKING	X.STK_L WW *nn Y.STK_L WW *nn Z.STK_L WW *nn	C210 [oper. 16 bits] C211 [oper. 16 bits] C212 [oper. 16 bits]	C210000B = X.STK.L WW11
RELATIVE	X.REL_L WW *nn Y.REL_L WW *nn Z.REL_L WW *nn B.REL_L WW *nn C.REL_L WW *nn	C220 [oper. 16 bits] C221 [oper. 16 bits] C222 [oper. 16 bits] C223 [oper. 16 bits] C224 [oper. 16 bits]	C2200041 = X.REL.L WW65
CHECKING	X.CTL_L WW *nn Y.CTL_L WW *nn Z.CTL_L WW *nn B.CTL_L WW *nn C.CTL_L WW *nn	C230 [oper. 16 bits] C231 [oper. 16 bits] C232 [oper. 16 bits] C233 [oper. 16 bits] C234 [oper. 16 bits]	C2300010 = X.CTL.L WW16
<u>ROTATING</u>			
ABSOLUTE *(Words)*	X.ABS_R WW *nn Y.ABS_R WW *nn Z.ABS_R WW *nn B.ABS_R WW *nn C.ABS_R WW *nn	C300 [oper. 16 bits] C301 [oper. 16 bits] C302 [oper. 16 bits] C303 [oper. 16 bits] C304 [oper. 16 bits]	C300000A = X.ABS.R WW10
STACKING	X.STK_R WW *nn Y.STK_R WW *nn Z.STK_R WW *nn *(nn = 00 to 55 and 64 to 65)	C310 [oper. 16 bits] C311 [oper. 16 bits] C312 [oper. 16 bits]	C3100020 = X.STK.R WW32

Type of Instruction	Display	Codop (hexadecimal)	Examples
RELATIVE	X.REL_R WW *nn Y.REL_R WW *nn Z.REL_R WW *nn B.REL_R WW *nn C.REL_R WW *nn	C320 [oper. 16 bits] C321 [oper. 16 bits] C322 [oper. 16 bits] C323 [oper. 16 bits] C324 [oper. 16 bits]	C3200001 = X.REL.R WW01
CHECKING	X.CTL_R WW *nn Y.CTL_R WW *nn Z.CTL_R WW *nn B.CTL_R WW *nn C.CTL_R WW *nn *(nn = 00 to 55 and 64 to 65)	C330 [oper. 16 bits] C331 [oper. 16 bits] C332 [oper. 16 bits] C333 [oper. 16 bits] C334 [oper. 16 bits] WWORD No.	C3300041 = X.CTL.R WW65
FREE	X. FREE Y. FREE Z. FREE B. FREE C. FREE	C040 C041 C042 C043 C044	
LINE	LIN.	B046	

Type of Instruction	Display	Codop (hexadecimal)	Examples
<u>**LINEAR**</u>			
POS_ANA	X = POS ANA + distance Y = POS ANA + distance Z = POS ANA + distance B = POS ANA + distance C = POS ANA + distance	C060 [oper. 32 bits] C061 [oper. 32 bits] C062 [oper. 32 bits] C063 [oper. 32 bits] C064 [oper. 32 bits]	
POS_NUM	X = POS NUM + distance Y = POS NUM + distance Z = POS NUM + distance B = POS NUM + distance C = POS NUM + distance	C070 [oper. 32 bits] C071 [oper. 32 bits] C072 [oper. 32 bits] C073 [oper. 32 bits] C074 [oper. 32 bits]	
VEL ANA NORMAL	X = VEL ANA_N + distance Y = VEL ANA_N + distance Z = VEL ANA_N + distance B = VEL ANA_N + distance C = VEL ANA_N + distance	C080 [oper. 32 bits] C081 [oper. 32 bits] C082 [oper. 32 bits] C083 [oper. 32 bits] C084 [oper. 32 bits]	
VEL ANA INTEGRAL	X = VEL ANA_I + distance Y = VEL ANA_I + distance Z = VEL ANA_I + distance B = VEL ANA_I + distance C = VEL ANA_I + distance	C090 [oper. 32 bits] C091 [oper. 32 bits] C092 [oper. 32 bits] C093 [oper. 32 bits] C094 [oper. 32 bits]	
VEL NUM NORMAL	X = VEL NUM_N + distance Y = VEL NUM_N + distance Z = VEL NUM_N + distance B = VEL NUM_N + distance C = VEL NUM_N + distance	C0A0[oper. 32 bits] C0A1 [oper. 32 bits] C0A2 [oper. 32 bits] C0A3 [oper. 32 bits] C0A4 [oper. 32 bits]	
VEL NUM INTEGRAL	X = VEL NUM_I + distance Y = VEL NUM_I + distance Z = VEL NUM_I + distance B = VEL NUM_I + distance C = VEL NUM_I + distance	C0B0[oper. 32 bits] C0B1 [oper. 32 bits] C0B2 [oper. 32 bits] C0B3 [oper. 32 bits] C0B4 [oper. 32 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
<u>ROTATING</u>			
POS_ANA	X = POS ANA + angle Y = POS ANA + angle Z = POS ANA + angle B = POS ANA + angle C = POS ANA + angle	C160 [oper. 32 bits] C161 [oper. 32 bits] C162 [oper. 32 bits] C163 [oper. 32 bits] C164 [oper. 32 bits]	
POS_NUM	X = POS NUM + angle Y = POS NUM + angle Z = POS NUM + angle B = POS NUM + angle C = POS NUM + angle	C170 [oper. 32 bits] C171 [oper. 32 bits] C172 [oper. 32 bits] C173 [oper. 32 bits] C174 [oper. 32 bits]	
VEL ANA NORMAL	X = VEL ANA_N + angle Y = VEL ANA_N + angle Z = VEL ANA_N + angle B = VEL ANA_N + angle C = VEL ANA_N + angle	C180 [oper. 32 bits] C181 [oper. 32 bits] C182 [oper. 32 bits] C183 [oper. 32 bits] C184 [oper. 32 bits]	
VEL ANA INTEGRAL	X = VEL ANA_I + angle Y = VEL ANA_I + angle Z = VEL ANA_I + angle B = VEL ANA_I + angle C = VEL ANA_I + angle	C190 [oper. 32 bits] C191 [oper. 32 bits] C192 [oper. 32 bits] C193 [oper. 32 bits] C194 [oper. 32 bits]	
VEL NUM NORMAL	X = VEL NUM_N + angle Y = VEL NUM_N + angle Z = VEL NUM_N + angle B = VEL NUM_N + angle C = VEL NUM_N + angle	C1A0[oper. 32 bits] C1A1 [oper. 32 bits] C1A2 [oper. 32 bits] C1A3 [oper. 32 bits] C1A4 [oper. 32 bits]	
VEL NUM INTEGRAL	X = VEL NUM_I + angle Y = VEL NUM_I + angle Z = VEL NUM_I + angle B = VEL NUM_I + angle C = VEL NUM_I + angle	C1B0[oper. 32 bits] C1B1 [oper. 32 bits] C1B2 [oper. 32 bits] C1B3 [oper. 32 bits] C1B4 [oper. 32 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
<u>**TEST, CONDITIONS**</u>			
. 1 Operand			
on Bit	IF BIT 000 (to 127)	D000 [oper. 16 bits]	
	IF/BIT 000 (to 127)	D010 [oper. 16 bits]	
on Output	IF OUT 000 (to 255)	D001 [oper. 16 bits]	
	IF/OUT 000 (to 255)	D011 [oper. 16 bits]	
on Input	IF IN/000 (to 255)	D002 [oper. 16 bits]	
	IF IN 000 (to 255)	D003 [oper. 16 bits]	
	IF/IN 000 (to 255)	D013 [oper. 16 bits]	
on Timer	IF TIM 00 (to 15)	D004 [oper. 16 bits]	
	IF/TIM 00 (to 15)	D014 [oper. 16 bits]	
		Operand No.	
. 2 Operands			
* on Word (16 bits)	**IF WRD 000** (to 4095)	D300 [oper. 16 bits]	
–> 1st Operand	IF/WRD 000 (to 4095)	D310 [oper. 16 bits]	
with decimal value	= 0000 (to 9999)	D400 [oper. 16 bits]	***Note*** : If the decimal value cannot exceed 9,999, the hexadecimal value goes up to 65,535.
	> = 0000 (to 9999)	D401 [oper. 16 bits]	
	< = 0000 (to 9999)	D402 [oper. 16 bits]	
	AND 0000 (to 9999)	D403 [oper. 16 bits]	
with hexadecimal value	= 0000 (to FFFF)	D410 [oper. 16 bits]	
	> = 0000 (to FFFF)	D411 [oper. 16 bits]	
	< = 0000 (to FFFF)	D412 [oper. 16 bits]	
	AND 0000 (to FFFF)	D413 [oper. 16 bits]	
with Counter	= CNT 00 (to 15)	D420 [oper. 16 bits]	
	> = CNT 00 (to 15)	D421 [oper. 16 bits]	
	< =CNT 00 (to 15)	D422 [oper. 16 bits]	
	AND CNT 00 (to 15)	D423 [oper. 16 bits]	
with Inputs (modulo 16)	=IN 000 (to 112)	D430 [oper. 16 bits]	
	> =IN 000 (to 112)	D431 [oper. 16 bits]	
	< =IN 000 (to 112)	D432 [oper. 16 bits]	
	AND IN 000 (to 112)	D433 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095)	D440 [oper. 16 bits]	
	> = WRD 0000 (to 4095)	D441 [oper. 16 bits]	
	< = WRD 0000 (to 4095)	D442 [oper. 16 bits]	
	AND WRD 0000(to 4095)	D443 [oper. 16 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
* on WWord (32 bits) –> 1st Operand	**IF WWRD** 000 (to 127) IF/WWRD 000 (to 127)	D320 [oper. 16 bits] D330 [oper. 16 bits]	
with decimal value	= 00000000 (to 09999999) > = 00000000 (to 09999999) < = 00000000 (to 09999999) AND 00000000 (to 09999999)	D500 [oper. 32 bits] D501 [oper. 32 bits] D502 [oper. 32 bits] D503 [oper. 32 bits]	***Note*** : If the decimal value cannot exceed 9,999,999, the hexadecimal value goes up to 4,294,967,295.
with hexadecimal value	= 00000000 (to FFFFFFFF) > = 00000000 (to FFFFFFFF) < = 00000000 (to FFFFFFFF) AND00000000 (to FFFFFFFF)	D510 [oper. 32 bits] D511 [oper. 32 bits] D512 [oper. 32 bits] D513 [oper. 32 bits]	
with Counter	= CNT 00 (to 15) > = CNT 00 (to 15) < = CNT 00 (to 15) AND CNT 00 (to 15)	D520 [oper. 16 bits] D521 [oper. 16 bits] D522 [oper. 16 bits] D523 [oper. 16 bits]	
with Inputs (modulo 16)	= IN 000 (to 112) > = IN 000 (to 112) < = IN 000 (to 112) AND IN 000 (to 112)	D530 [oper. 16 bits] D531 [oper. 16 bits] D532 [oper. 16 bits] D533 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095) > = WRD 0000 (to 4095) < = WRD 0000 (to 4095) AND WRD 0000(to 4095)	D540 [oper. 16 bits] D541 [oper. 16 bits] D542 [oper. 16 bits] D543 [oper. 16 bits]	
with WWord (32 bits)	= WWRD 000 (to 127) > = WWRD 000 (to 127) < = WWRD 000 (to 127) AND WWRD 000(to 127)	D550 [oper. 16 bits] D551 [oper. 16 bits] D552 [oper. 16 bits] D553 [oper. 16 bits]	
* on Counter –> 1st Operand	**IF CNT** 00 (to 15) IF/CNT 00 (to 15)	D340 [oper. 16 bits] D350 [oper. 16 bits]	
with decimal value	= 0000 (to 9999) > = 0000 (to 9999) < = 0000 (to 9999) AND 0000 (to 9999)	D900 [oper. 16 bits] D901 [oper. 16 bits] D902 [oper. 16 bits] D903 [oper. 16 bits]	
with hexadecimal value	= 0000 (to FFFF) > = 0000 (to FFFF) < = 0000 (to FFFF) AND 0000 (to FFFF)	D910 [oper. 16 bits] D911 [oper. 16 bits] D912 [oper. 16 bits] D913 [oper. 16 bits]	
with Counter	= CNT 00 (to 15) > = CNT 00 (to 15) < = CNT 00 (to 15) AND CNT 00 (to 15)	D920 [oper. 16 bits] D921 [oper. 16 bits] D922 [oper. 16 bits] D923 [oper. 16 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
with Inputs (modulo 16)	= IN 000 (to 112) > = IN 000 (to 112) < = IN 000 (to 112) AND IN 000 (to 112)	D930 [oper. 16 bits] D931 [oper. 16 bits] D932 [oper. 16 bits] D933 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095) > = WRD 0000 (to 4095) < = WRD 0000 (to 4095) AND WRD 0000(to 4095)	D940 [oper. 16 bits] D941 [oper. 16 bits] D942 [oper. 16 bits] D943 [oper. 16 bits]	
INITIALIZATION **. 1 Operand**			
* on Bit –> 1 on Bit –> 0	SET.BIT 032 (to 127) RST.BIT 032 (to 127)	D015 [oper. 16 bits] D017 [oper. 16 bits]	
* on Output –> 1 on Output –> 0	SET.OUT 000 (to 127) RST.OUT 000 (to 127)	D016 [oper. 16 bits] D018 [oper. 16 bits]	
* on Word –> 0	RST.WRD 0000 (to 4095)	D019 [oper. 16 bits] Variable number	
* on WWord –> 0	RST.WWRD 00 (to 63)	D01D [oper. 16 bits] Variable number	
* on Counter –> 0	RST.CNT 0000 (to 0015)	D01A 00 [oper. 8 bits] Counter number	
	RST.CNT 0041 (to 9980)	D01A[oper. 8 bits] [oper. 8 bits] PRG No. SP No.	
. 2 Operands			
* on Word (16 bits) –> 1st Operand	**SET.WRD** 0000 (to 4095)	D600 [oper. 16 bits]	
with decimal value	= 0000 (to 9999) + 0000 (to 9999) – 0000 (to 9999) x 0000 (to 9999) / 0000 (to 9999) AND 0000 (to 9999) OR 0000 (to 9999)	D700 [oper. 16 bits] D701 [oper. 16 bits] D702 [oper. 16 bits] D703 [oper. 16 bits] D704 [oper. 16 bits] D705 [oper. 16 bits] D706 [oper. 16 bits]	
with hexadecimal value	= 0000 (to FFFF) + 0000 (to FFFF) – 0000 (to FFFF) x 0000 (to FFFF) / 0000 (to FFFF) AND 0000 (to FFFF) OR 0000 (to FFFF)	D710 [oper. 16 bits] D711 [oper. 16 bits] D712 [oper. 16 bits] D713 [oper. 16 bits] D714 [oper. 16 bits] D715 [oper. 16 bits] D716 [oper. 16 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
with Counter	= CNT 00 (to 15) + CNT 00 (to 15) – CNT 00 (to 15) x CNT 00 (to 15) / CNT00 (to 15) AND CNT 00 (to 15) OR CNT 00 (to 15)	D720 [oper. 16 bits] D721 [oper. 16 bits] D722 [oper. 16 bits] D723 [oper. 16 bits] D724 [oper. 16 bits] D725 [oper. 16 bits] D726 [oper. 16 bits]	
with Inputs (modulo 16)	= IN 000 (to 112) + IN 000 (to 112) – IN 000 (to 112) x IN 000 (to 112) / IN 000 (to 112) AND IN 000 (to 112) OR IN 000 (to 112)	D730 [oper. 16 bits] D731 [oper. 16 bits] D732 [oper. 16 bits] D733 [oper. 16 bits] D734 [oper. 16 bits] D735 [oper. 16 bits] D736 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095) + WRD 0000 (to 4095) – WRD 0000 (to 4095) x WRD 0000 (to 4095) / WRD 0000 (to 4095) AND WRD 0000 (to 4095) OR WRD 0000 (to 4095)	D740 [oper. 16 bits] D741 [oper. 16 bits] D742 [oper. 16 bits] D743 [oper. 16 bits] D744 [oper. 16 bits] D745 [oper. 16 bits] D746 [oper. 16 bits]	
* on WWord (32 bits) –> 1st Operand	**SET.WWRD 000** (to 127)	D620 [oper. 16 bits]	
with decimal value	= 00000000 (to 09999999) + 00000000 (to 09999999) – 00000000 (to 09999999) x 00000000 (to 09999999) / 00000000 (to 09999999) AND 00000000 (to 09999999) OR 00000000 (to 09999999)	D800 [oper. 32 bits] D801 [oper. 32 bits] D802 [oper. 32 bits] D803 [oper. 32 bits] D804 [oper. 32 bits] D805 [oper. 32 bits] D806 [oper. 32 bits]	
with hexadecimal value	= 00000000 (to FFFFFFFF) + 00000000 (to FFFFFFFF) – 00000000 (to FFFFFFFF) x 00000000 (to FFFFFFFF) / 00000000 (to FFFFFFFF) AND 00000000 (to FFFFFFFF) OR 00000000 (to FFFFFFFF)	D810 [oper. 32 bits] D811 [oper. 32 bits] D812 [oper. 32 bits] D813 [oper. 32 bits] D814 [oper. 32 bits] D815 [oper. 32 bits] D816 [oper. 32 bits]	
with Counter	= CNT 00 (to 15) + CNT 00 (to 15) – CNT 00 (to 15) x CNT 00 (to 15) / CNT 00 (to 15) AND CNT 00 (to 15) OR CNT 00 (to 15)	D820 [oper. 16 bits] D821 [oper. 16 bits] D822 [oper. 16 bits] D823 [oper. 16 bits] D824 [oper. 16 bits] D825 [oper. 16 bits] D826 [oper. 16 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
with Inputs (modulo 16)	= IN *nn	D830 [oper. 16 bits]	
	+ IN *nn	D831 [oper. 16 bits]	
	– IN *nn	D832 [oper. 16 bits]	
*nn = 00 to 112	x IN *nn	D833 [oper. 16 bits]	
and 136 to 240	/ IN *nn	D834 [oper. 16 bits]	
	AND IN *nn	D835 [oper. 16 bits]	
	OR IN *nn	D836 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095)	D840 [oper. 16 bits]	
	+ WRD 0000 (to 4095)	D841 [oper. 16 bits]	
	– WRD 0000 (to 4095)	D842 [oper. 16 bits]	
	x WRD 0000 (to 4095)	D843 [oper. 16 bits]	
	/ WRD 0000 (to 4095)	D844 [oper. 16 bits]	
	AND WRD 0000 (to 4095)	D845 [oper. 16 bits]	
	OR WRD 0000 (to 4095)	D846 [oper. 16 bits]	
with WWord (32 bits)	= WWRD *nn and 200–202	D850 [oper. 16 bits]	
	+ WWRD *nn	D851 [oper. 16 bits]	
	– WWRD *nn	D852 [oper. 16 bits]	
*nn = 0 to 127	x WWRD *nn	D853 [oper. 16 bits]	
	/ WWRD *nn	D854 [oper. 16 bits]	
	AND WWRD*nn	D855 [oper. 16 bits]	
	OR WWRD *nn	D856 [oper. 16 bits]	
* on Counter	**SET.CNT** 0000 (to 0015)	D640 [oper. 8 bits]	Standard counter
–> 1st Operand	SET.CNT 0041 (to 9980)	D640[oper. 8 bits] [oper. 8 bits] PRG No. SP No.	Stacking counter
with decimal value	= 0000 (to 9999)	DA00 [oper. 16 bits]	
	+ 0000 (to 9999)	DA01 [oper. 16 bits]	
	– 0000 (to 9999)	DA02 [oper. 16 bits]	
	x 0000 (to 9999)	DA03 [oper. 16 bits]	
	/ 0000 (to 9999)	DA04 [oper. 16 bits]	
	AND 0000 (to 9999)	DA05 [oper. 16 bits]	
	OR 0000 (to 9999)	DA06 [oper. 16 bits]	
with hexadecimal value	= 0000 (to FFFF)	DA10 [oper. 16 bits]	
	+ 0000 (to FFFF)	DA11 [oper. 16 bits]	
	– 0000 (to FFFF)	DA12 [oper. 16 bits]	
	x 0000 (to FFFF)	DA13 [oper. 16 bits]	
	/ 0000 (to FFFF)	DA14 [oper. 16 bits]	
	AND 0000 (to FFFF)	DA15 [oper. 16 bits]	
	OR 0000 (to FFFF)	DA16 [oper. 16 bits]	
with Counter	= CNT 00 (to 15)	D920 [oper. 16 bits]	
	+ CNT 00 (to 15)	D921 [oper. 16 bits]	
	– CNT 00 (to 15)	D922 [oper. 16 bits]	
	x CNT 00 (to 15)	D922 [oper. 16 bits]	
	/ CNT 00 (to 15)	D922 [oper. 16 bits]	
	AND CNT 00 (to 15)	D923 [oper. 16 bits]	
	OR CNT 00 (to 15)	D923 [oper. 16 bits]	

Type of Instruction	Display	Codop (hexadecimal)	Examples
with Inputs (modulo 16)	= IN 000 (to 112) + IN 000 (to 112) – IN 000 (to 112) x IN 000 (to 112) / IN 000 (to 112) AND IN 000 (to 112) OR IN 000 (to 112)	DA30 [oper. 16 bits] DA31 [oper. 16 bits] DA32 [oper. 16 bits] DA33 [oper. 16 bits] DA34 [oper. 16 bits] DA35 [oper. 16 bits] DA36 [oper. 16 bits]	
with Word (16 bits)	= WRD 0000 (to 4095) + WRD 0000 (to 4095) – WRD0000 (to 4095) x WRD 0000 (to 4095) / WRD 0000 (to 4095) AND WRD 0000 (to 4095) OR WRD 0000 (to 4095)	DA40 [oper. 16 bits] DA41 [oper. 16 bits] DA42 [oper. 16 bits] DA43 [oper. 16 bits] DA44 [oper. 16 bits] DA45 [oper. 16 bits] DA46 [oper. 16 bits]	
–> + 1	INC.CNT 0000 (to 0015) INC.CNT 0041 (to 9980)	D01B 00 [oper. 8 bits] (↓ Standard No.) D01B[oper. 8 bits] [oper. 8 bits] (↓ PRG No., ↓ SP No.)	
–> – 1	DEC.CNT 0000 (to 0015) DEC.CNT 0041 (to 9980)	D01C 00 [oper. 8 bits] (↓ Standard No.) D01C[oper. 8 bits] [oper. 8 bits] (↓ PRG No., ↓ SP No.)	

II – 2. PLC programs

Type of Instruction	Display	Codop (hexadecimal)
PROG.PLC xx header (num)	PLC xx	FC [oper. 16 bits] ↓ PLC No.
TEST CONDITION	IF ...	See part programs
INITIALISATION	SET ... RST ... INC ... DEC ...	See part programs
COMPARISON xxxx > = xxxx	CMP 0000 (to 0015) VAL 0000 (to FFFF) 0000 (to 0015)	D020 [oper. 16 bits] [oper. 16 bits] ↓ Counter No. ↓ Value
TIMER xx VALUE xxxx	TIMER 00 (to 15) VAL 0000 (to 9999)	D021 [oper. 16 bits] [oper. 16 bits] ↓ Timer No. ↓ Pre–selection No.
AND FUNCTION on BIT	AND BIT 000 (to 127)	D022 [oper. 16 bits]
AND FUNCTION on OUTPUT	AND OUT 000 (to 127)	D023 [oper. 16 bits]
AND FUNCTION on BIT	OR BIT 000 (to 127)	D024 [oper. 16 bits]
OR FUNCTION on OUTPUT	OR OUT 000 (to 127)	D025 [oper. 16 bits] ↓ Variables No.
END OF PROGRAM	END	F5 [oper. 16 bits] ↓ PLC No.

III – PROGRAM CODES

III – 1. Declaration of programs, subroutines and PLCs

▶ Header codes of PRG, SP,..., SR, PLC

- F9b xn = Main program
- b = 0, standard PRG (encoded on 15 bits)
  b = 1 , SAP PRG (encoded on 15 bits)
- FAnn = STD, STK.. // subroutine (see stacking header)
- FBnn = Return subroutine (see home return header)
- FCnn = PLC program
- FEnn = FREE

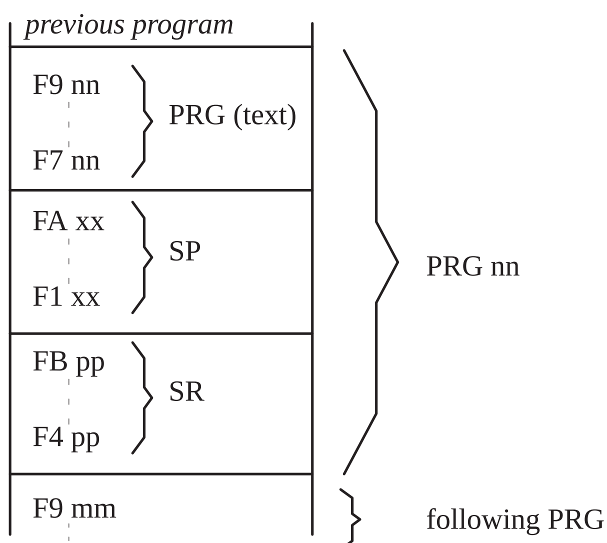
▶ STEP TRANSITION codes

- EC00 + Step number 0 to 999
- E.g. : EC12 => Step number 18 (decimal)
- E.g. : ED00 => Step number 256 (decimal)

▶ END of PRG, SP..., SR, PLC codes

- F0nn = End of ”standard” SP nn.
- F1nn = End of ”standard” stacking SP nn.
- F2nn = End of ”general” stacking SP nn.
- F3nn = End of SP // nn.
- F4nn = End of simple or total SR nn.
- F8nn = End of simple or total SR with return to step 0 of PRG 00.
- F5nn = End of PLC nn.
- F7nn = End of main program (PRG) nn.

▶ PRG architecture in the memory area



III – 2. Subroutine and program calls

▶ SPECIFIC codes for SP, SR, PLC as an instruction

- E000 [oper. 16 bits] :

Standard SP SP nn Lmm (nn = 01 to 40) (mm = 00 to 99)

Regular Stacking SP SP nn D Lmm (or I Lmm) (nn = 41 to 60) (mm = 00 to 99)

General Stacking SP SP nn D Lmm (or I Lmm) (nn = 61 to 80) (mm = 00 to 99)

Parallel SP SP nn L00 (nn = 81 to 99)

The operand contains :

. high order word –> the LABEL number

–> bit 0 x 8000 at 0 indicates DIRECT

–> bit 0 x 8000 at 1 indicates REVERSE

. low order word –> the SP number.

E.g. : E000 0103 –> SP 03 L01

E.g. : E000 8229 –> SP 41 I L02

- E100 [oper. 16 bits] : PLC prog. – Display : PLC 00 (to 99)
- E500 [oper. 16 bits] : Home Return – Display : SR 01 (to 99)

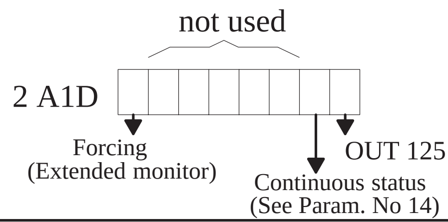
▶ Return label

- E600 [oper. 16 bits] : Labels ”L” for SP – Display : L00 to L99
- E700 [oper. 16 bits] : Labels ”R” for SR – Display : R00 to R99

IV – VARIABLE ADDRESSING

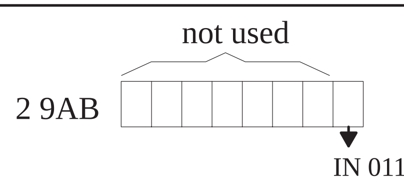
IV – 1. <u>Output – OUT –</u>

Accessible in read and write.

Number (logical address)	Physical address	Structures / Functions
OUT 000 ↓ OUT 255	28A0 ↓ 299F	

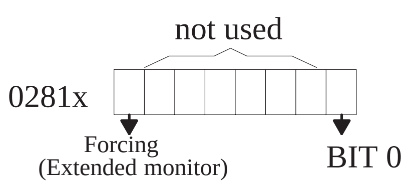
IV – 2. <u>Input – IN –</u>

Accessible in read.

Number (logical address)	Physical address	Structures / Functions
IN 000 ↓ IN 255	29A0 ↓ 2A9F	

IV – 3. <u>User and system bits – BIT –</u>

Each address corresponds to an 8 bit structure in memory.



x = bit number in hexadecimal (e.g.: Bit 31, address = 0282F).
Only the low order word is used.

– System bits accessible in Read – No. 0 to 30.

– System bits accessible in Read and Write – No. 31 to 33.

– User bits accessible in Read and Write – No. 34 to 127.

For the definition of these bits, see the Programming Level 2 manual, paragraph I3.

IV – 4. 16 bits user and system words – WRD –

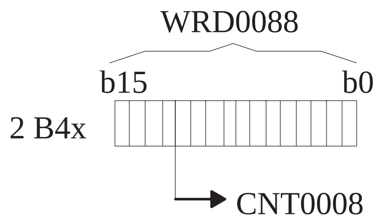
Number (logical address)	Physical address	Structures / Functions
WRD 0000 ↓ WRD 0031	2AA0 ↓ 2ADF	32 user Words (read/write) with no predefined functions. B15 0 16 bit structure available
WRD 0032 ↓ WRD 0063	2AE0 ↓ 2B1E	32 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I4
WRD 0064 ↓ WRD 0079	2B20 ↓ 2B3F	16 user Words (read/write) supporting the PLC timers (TIM 00 to TIM 15).
WRD 0080 ↓ WRD 0095	2B40 ↓ 2B5F	16 user Words (read/write) supporting the standard counters (CNT 00 to CNT 15).
WRD 0096 WRD 4096	2B60 3A9F	4000 user Words (read/write) supporting the stacking subroutine counters (CNT 0041 to CNT 9980).

IV – 5. 32 bit user and system words – WWRD –

Number (logical address)	Physical address	Structures / Functions
WWRD 000 ↓ WWRD 063	6230 ↓ 6327	64 user Words (read/write) with no predefined functions. b31 0 32 bit structure available
WWRD 064 ↓ WWRD 127	6328 ↓ 642C	64 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I5
WWRD 0116 WWRD 0117	6400 6404	*Specific words* Values for calculating the automatic anticipated restart. Values for calculating the automatic anticipated restart. See chapter VI – page 28.

IV – 6. Counters

Each address corresponds to a 16 bit structure in the memory.



. values from 0000 to 9999 in decimal

. values from 0000 to FFFF in hexadecimal

x = bit number in hexadecimal (e.g.: CNT 0008, address = 2 B50).

– Standard counters – No. 0000 to 0015 (0x2B40 to 0x2B5E).

– Regular stacking counters – No. 0041 to 9960 (as from 0x2 B60).

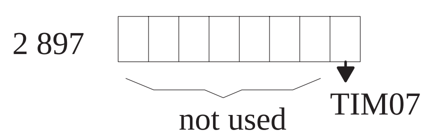
– General stacking counters – No 0061 to 9980.

For the definition of these counters, see the Programming Level 2 manual, paragraph I6.

IV – 7. Timers

IV – 7. 1.End of timer for part program

Accessible in read and write.

Number (logical address)	Physical address	Structures / Functions
TIM00	2 890	 Only the low order word is used
TIM01	2 891	
TIM02	2 892	
TIM03	2 893	
TIM04	2 894	
TIM05	2 895	
TIM06	2 896	
TIM07	2 897	
TIM08	2 898	
TIM09	2 899	
TIM10	2 89A	
TIM11	2 89B	
TIM12	2 89C	
TIM13	2 89D	
TIM14	2 89E	
TIM15	2 89F	

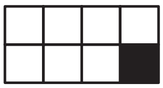
IV – 7. 2.PLC timer

TIM00 to 15 = WRD 0064 to 0079 see chapter IV – 4.
Accessible in read and write.

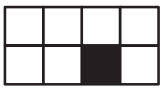
V – CPU FAULT SIGNALLING

V – 1. Flashing Leds

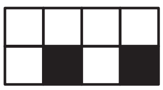
These signal a CAN network fault by displaying the problem number in binary on the LEDs at the bottom of the CPU, and the node number (if concerned) on the LEDs at the top if the pendant is not functioning.


3 0

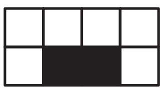
1 = CAN driver initialization fault



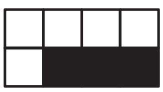
2 = Write problem in Flashprom



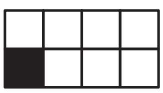
5 = A double (or more) node on the network (code + node)



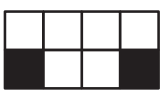
6 = Problem during the CONNECTION phase (code + node)



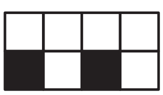
7 = Problem during the PREPARATION phase (code + node)



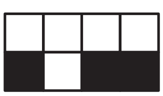
8 = Problem during the START phase (code + node)



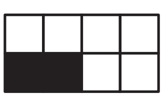
9 = The network does not correspond to the parametered configuration (code + node)



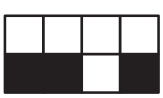
10 = “Node–guarding” problem (code + node). Communication fault with the pendant ; this may be due to the CAN speed being too great for the length of the cable used, or a bad line adaptation, or interference, etc.



11 = CPU emission problem



12 = CPU reception problem



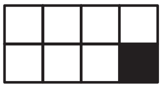
13 = Topology fault of the remote I/O

15 = EMERGENCY message received (code + node). Problem on the pendant or with communication between the pendant and the CPU (see 10)

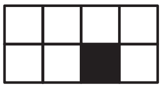
Note : In the event of a NODE GUARDING fault, fault 15 may appear alternately with fault 10.

V – 2. <u>Fixed Leds</u>

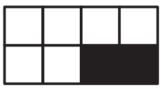
These signal a fault when powering up by giving the problem number in binary on the LEDs at the bottom of the CPU, and the node number (if concerned) on the LEDs at the top if the pendant is not functioning.



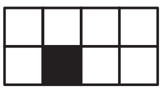
1 = Problem with recovering the parameters in Flashprom



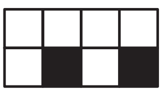
2 = Problem during the opening of the PC link



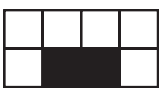
3 = Problem during the opening of the EUROMAP 17 link



4 = Problem during the opening of the printer 2 link

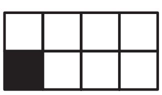


5 = Problem during the opening of the CAN link

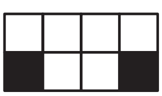


6 = Message not present in Flashprom

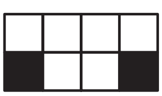
7 = Problem with the CPU’s RAM



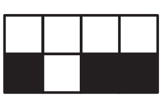
8 = Problem with the Flashprom’s checksum



9 = Problem with the axes declared and the axes’ boards present



10 = The configuration has changed



11 = Problem during the initialization of the axes’ boards by the CPU

15 = Communication problem with the pendant during powering up. The CAN speed may be changed by transfering the parameters with the PC at 2400 Bds, slave = 1.

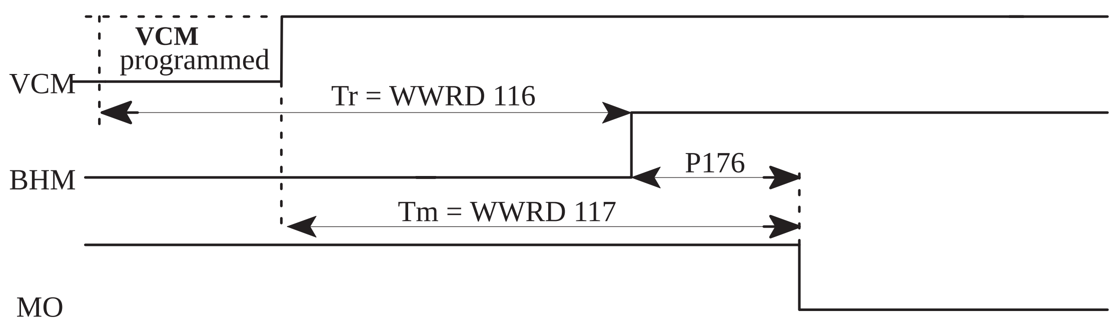
VI – IMM ANTICIPATED RESTART

- Parameter 174 : type of IMM anticipated restart
  - 0 : no anticipated restart
  - 1 : anticipated restart
  - 2 : programmed delay anticipated restart –> WWRD 63 programmed in step 0.
- Parameter 175 : basic value of the auto–adaptative delay and double the minimum value of the programmed delay
- Parameter 176 : minimum value of the auto–adaptative delay (safety margin)

Anticipated restart effective if :

- offset wait is not valid (parameter 451)
- and if the robot is in automatic
- and if Kv equals 100 %
- and if there is a SET WWRD63 in step 0 of the program
- and if the value of WWRD63 is greater than or equal to $\frac{\text{parameter 175}}{2}$

} in the case of restart with programmed delay

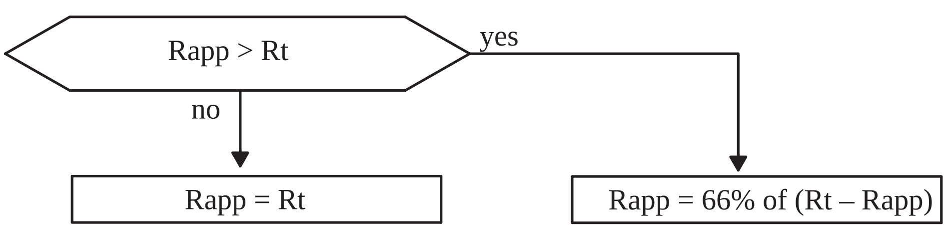


Tr = robot disengaging time in 1/10 s (WWRD 116)

Tm = IMM motion start time in 1/10 s (WWRD 117)

Rt = theoretical delay = Tr – Tm + P176 or 0 if the result is negative

Rapp = Applied delay



There is a fault if mould open (or OPA) goes to 0 and BHM = 0

`D_5 : MOVEMENT OUTSIDE CAMS` (if there is no anticipated restart running)

`D_32: PREMATURE MACHINE RESTART` (if there is an anticipated restart running)

▶ Safety circuit principle.

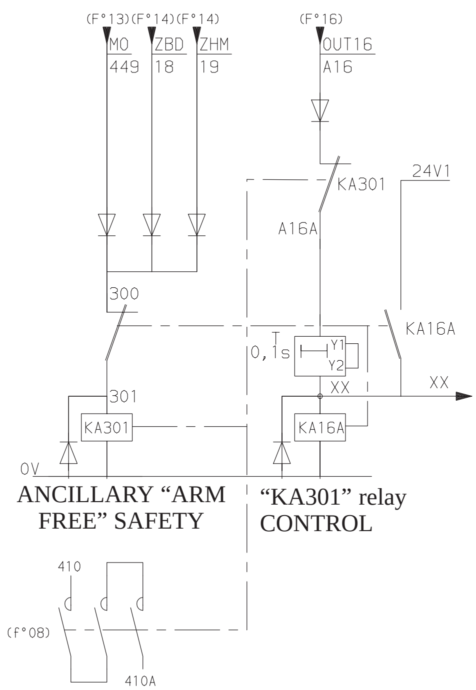
A hard–wired circuit controls the respective positions of the moving mould (“MO” = Mould Open signal) and of the robot (“ZBD” = Arm Free Area / “ZHM” = Outside Mould Area signal).

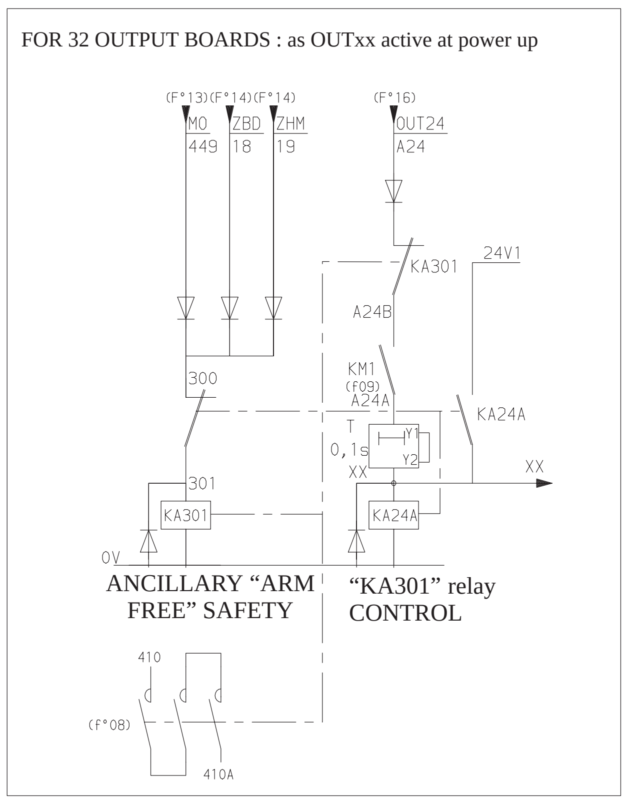
The output of this hard–wired circuit (”MO” + ”ZBD” + ”ZHM” = ”KA301”) activates a power relay (KA301 contactor).

During normal operation, the KA301 relay is activated. The KA301 contacts are used in series with the SBD relay contact from the interface board, which therefore means that the software safety that manages the SBD relay with a hard–wired safety device is doubled.

When there is a fault (robot position not conform compared to the moving mould position), the KA301 relay falls, which in turn activates the control relay KA16A, which is self–powered and which stops the KA301 relay becoming active (the blocking of KA301 prohibits the IMM cycle).

You must power down the robot cabinet to cancel this fault.





IF IN XX
SET WORD 62 = 200

Until a parameter for the control input for the anticipated restart safety circuit is integrated into the software, this input must be monitored and a fault must be generated using the monitoring PLC.

RELANCE ANTICIPEE NON CONFORME : in French

ANTICIPATED RESTART NOT CONFORM : in English

REARME ANTICIPADO NO CONFORME : in Spanish

VORAUSB. NEUSTART FEHLERHAFT : in German